# GITHUB IMPLEMENTATION OF YOUR PAPER

<https://github.com/AgrawalAmey/safe-explorer>

<https://github.com/AgrawalAmey/safe-explorer/tree/master/safe_explorer/safety_layer>

# SAFETY AWARE RL NOTES FROM LECTURE

<https://gatech.instructure.com/courses/240434/files?preview=31637531>

<https://bluejeans.com/playback/s/YCbmGHxOGF5CDBtNOk1mx3vupgbQiwiYb5eRFBu6AyA8T7Y3CwpkQW7D2yPygjPy>

[4-6-safe-rl](https://docs.google.com/document/d/1_bGILBes9DBztp-RU1T0gnjjLCvJFpsLn2VwTrXHgkE/edit)

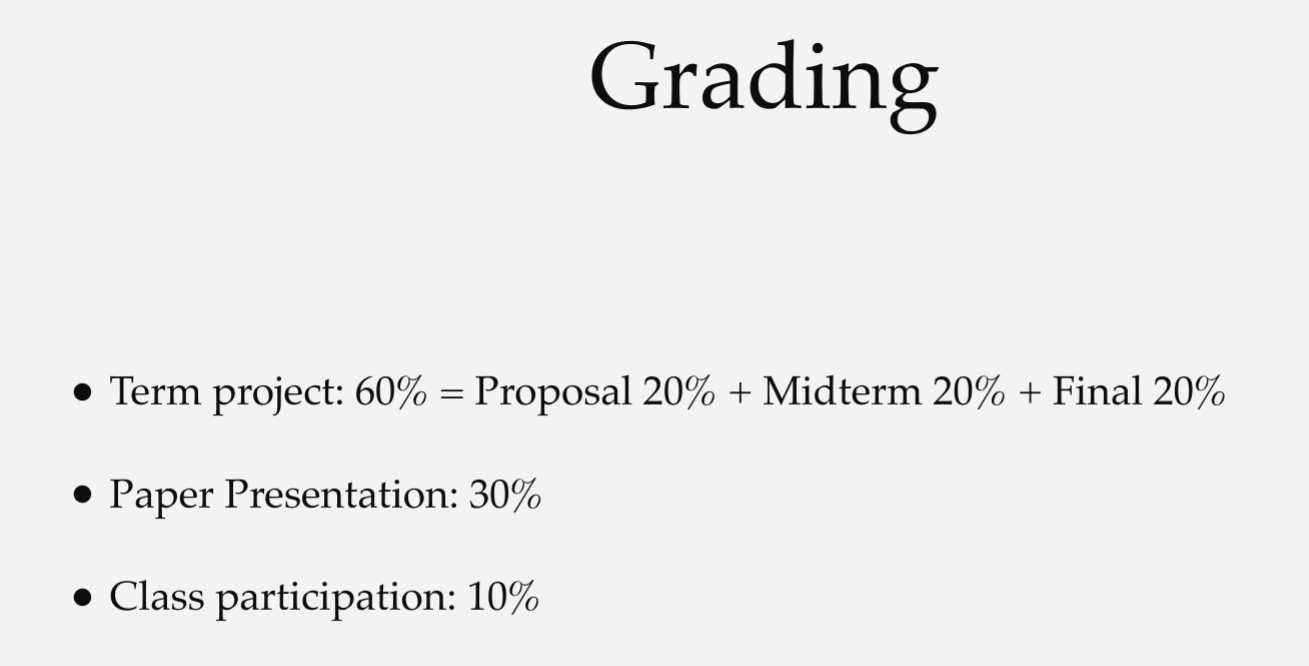
# THIS DOES ACTUALLY HAVE STUFF ON YOUR PAPER

<https://medium.com/@harshitsikchi/towards-safe-reinforcement-learning-88b7caa5702e>

Paper they kinda based all their stuff on

<https://arxiv.org/abs/1709.07643>

Evans, Richard and Gao, Jim. Deepmind ai reduces google data center cooling bill by 40%. DeepMind blog, 20, 2016.



<https://gatech.instructure.com/courses/240434/files?preview=28778983>

LECTURE 02 SLIDES THIS HAS THE THE EXPECTATIONS FOR THE PRESENTATION

We will have two paper presentations per lecture.

12 minutes presentation + 10 minutes discussion.

* Please include two to three quizzes.
* Please prepare one or two discussion points.
* Please submit the pdf copy to Canvas after the presentation.

Tips and what to include

* The rough structure can be: Motivation, Related Work, Method, Key Results, and Conclusion.
* Don’t be too concise or too descriptive. Best if each slide conveys a single idea and supporting materials.
* Don’t simply copy-and-paste the plots. Indicate what to see.

Eval criterion

* Understanding (50): does the presenter deeply understand the materials?
* Clarity (50): do the slides explain the ideas clearly?
* Completeness (50): do the slides present all the important ideas in the paper?
* Engagement (50): is the presentation interactive and engaging?
* Overall (100): overall impression from the instructor(s)
* Understanding: does the presenter clearly understand the materials?
* Clarity: is the paper presented clearly?
* Completeness: is the paper not missing any important ideas?
* Engagement: are the presentation and discussion engaging?

Paper #4: Dalal, G., Dvijotham, K., Vecerík, M., Hester, T., Paduraru, C., & Tassa, Y. (2018). Safe Exploration in Continuous Action Spaces. ArXiv, abs/1801.08757.

<https://www.semanticscholar.org/paper/Safe-Exploration-in-Continuous-Action-Spaces-Dalal-Dvijotham/7f567df97dc7e099d96e6c590ddf5aef8c5b11c4/video/da7365c2>

<https://www.youtube.com/watch?v=yr6y4Mb1ktI>

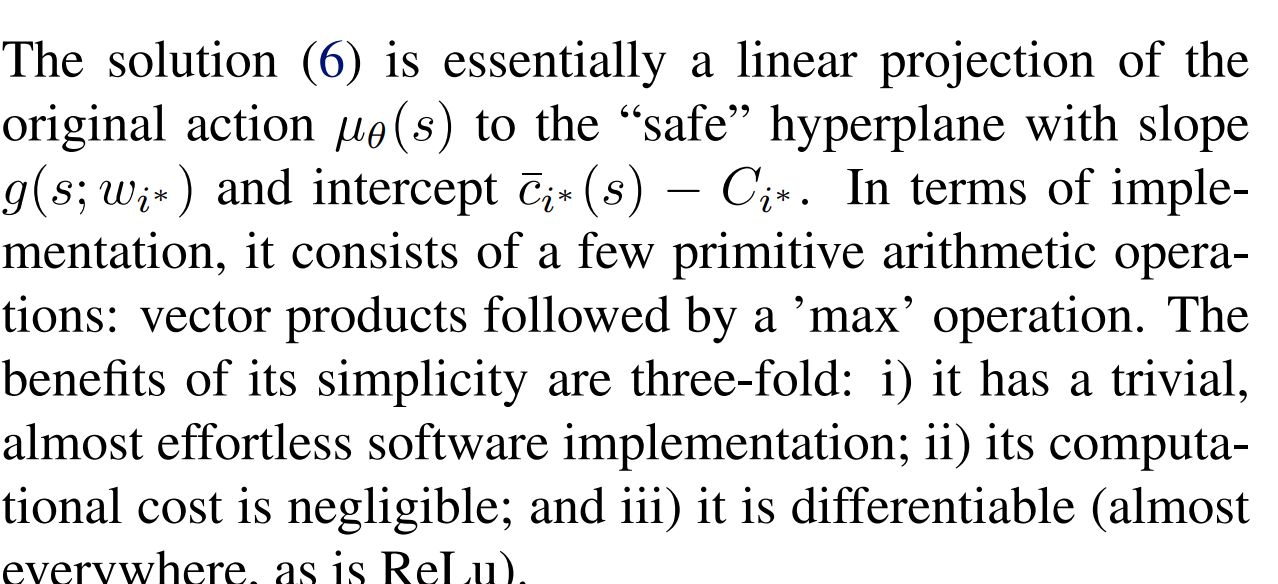
<https://www.google.com/search?q=Safe+Exploration+in+Continuous+Action+Spaces&sxsrf=APq-WBu8qlmoPxHqDyZLiQGeaFOnE3mzUA:1649695175165&source=lnms&tbm=vid&sa=X&ved=2ahUKEwiV1py3uYz3AhWfoWoFHbdsD4gQ_AUoBHoECAEQBg&biw=1920&bih=1011&dpr=1>

PRESENTATION NOTES

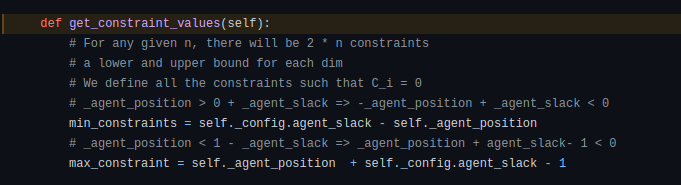
Our technique is to directly add to the policy a safety layer that analytically solves an action correction formulation per each state. The novelty of obtaining an elegant closed-form solution is attained due to a linearized model, learned on past trajectories consisting of arbitrary actions. This is to mimic the real-world circumstances where data logs were generated with a behavior policy that is implausible to describe mathematically; such cases render the known safety-aware off policy methods inapplicable. We demonstrate the efficacy of our approach on new representative physics-based environments, and prevail where reward shaping fails by maintaining zero constraint violations.

If you wanna do a slide on discrete vs continuous as a goal

Note that accomplishing this goal for discrete action spaces is more straightforward than for continuous ones. For instance, one can pre-train constraint-violation classifiers on offline data for pruning unsafe actions

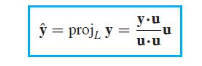


<https://github.com/AgrawalAmey/safe-explorer/blob/828673a0f8d6d41c3df05763707c141f93b3294e/safe_explorer/env/ballnd.py#L60>

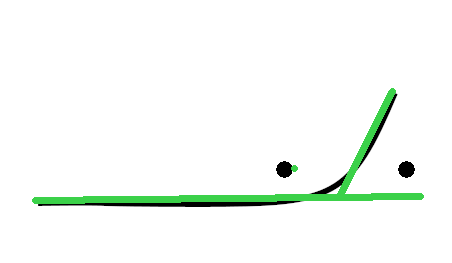


NOTE EVEN THOUGH THIS CAN FIGURE OUT FOR SOMETHING LIKE FORCE WHAT OUR CONSTRAINTS (POSITION) VALUE IS GOING TO BE IT CANT

THIS BREAKS FOR LARGE VALUES OF THE ACTION (Long rollouts in what the constraint would be if the constraint is nonlinear)



**EVEN IF WE HAVE NONLINEAR DATA WE ARE FITTING A LINEAR MODEL TO IT NOTE EVEN THOUGH WE SAY ITS LEARNED STATE WISE THE OVERALL ESTIMATION OF HOW THE CONSTRAINT SIGNAL CAN CHANGE GIVEN AND ACTION IS STILL LINEA SO FOR A DIAGRAM LIKE THE BELOW ITS GOING TO LEARN STRAIGHT LINE TO REPRESENT THIS CURVE OR WE HAVE TO BREAK IT UP INTO 2 CONSTRAINTS LIKE WE DO HERE**



* + Even with action inputs such as force we get a linear estimation of how the force will affect the position
* We never learn the whole transition function from this own

**Basically ask if give a g matrix and the knowledge you are moving left eg v\_x =-1, v\_y =0 given a formal problem setup which c\_i does that g matrix belong to.**

**When thinking about it in terms of a box where every corner is an obstacle. If you start in the center of the box and the action you set for a time step is to increase the velocity to the left (NOTE THE ACTION HAS TWO COMPONENTS x AND y IF WE ARE IN 2D)**

**Then you will grow closer to one boundary (left edge) but further from the right)**

What issues can you see in the comparison of this method and the others presented?

Reward shaping presents negative rewards and is being compared against methods that never present any negative reward

* Confused how the authors can even potentially claim that this is better on the notion of reward as they are literally using a technique that will induce many many more negative rewards. Notice one thing you see here is there are literally never any negative rewards on the blue or red BECAUSE THERE AREN'T ANY NEGATIVE REWARDS IN THE PROBLEM
  + Surprisingly, reward shaping poses no improvement but rather has an adverse effect: it resulted in highly negative episodic returns

Prior work

Constrained policy optimization(Achiam et al., 2017), proofs on safety were probabilistic (soft constraints)

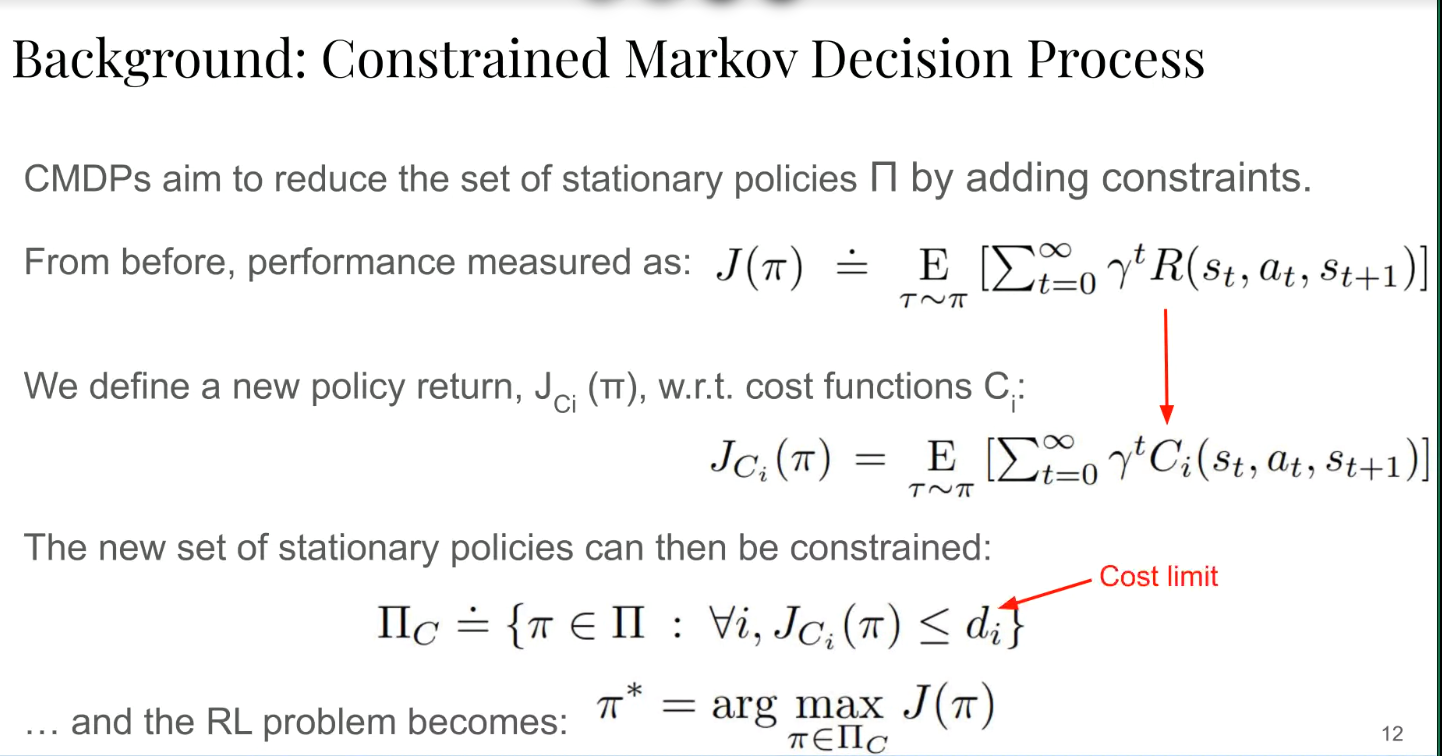
Safe model-based reinforcement learning with stability guarantees (Berkenkamp et al., 2017), to prove safety required prior knowledge of system

Optlayer-practical constrained optimization for deep reinforcement learning in the real world. (Pham et al., 2017), similar because it solved an optimization at the policy level on a state wise basis but relied on a very computationally expensive in-graph QP solver for updates. Also required prior knowledge of system

Goals of this method

To summarize, this work is the first, to our knowledge, to solve the problem of state-wise safety directly at the policy level, while also doing it in a data-driven fashion using arbitrary data logs. Moreover, it can be applied to any continuous-control algorithm; it is not restricted to a specific RL algorithm or any at all.

ADD REVIEW OF CMDPS SECTION AS THAT IS THE MOST PADDING SECTION YOU CAN GET



Zero constraint violations (hard constraint method)

Be able to train on data that was logged from various sources (not just those from a known behavior) which also allows the model to be pretrained so it can immediately enforce constraints when deployed

Does not require prior or expert knowledge of specific system

Implement as a safety layer that can be applied to any continuous control algorithm (RL or not

Safety signals

NOTE DURING THE EXPERIMENTS SECTION THEY SAY THIS WHICH MAKES ME THINK THE SAFETY SIGNAL IS ACTUALLY POSITION - BOUNDARY POSITION

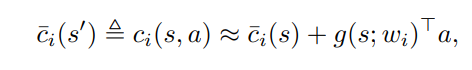
Each of the constraints, therefore, lower bounds the object’s distance to each of the few boundaries.

As the distance from each obstacle is modeled as a separate constraint, only a single obstacle is the closest one at a time. Proximity to corners shows to pose no issues, as can be seen in the plots and videos in Section 7

**REMEMBER**

**THIS WHOLE SETUP IS FOR c\_i which means THIS EQUATION IS DONE SEPARATELY FOR EACH CONSTRAINT**

**OH IMFUCKING STUPID THE WHOLE REASON G IS THE SAME DIMMS AS a IS BECAUSE ITS A FUCKING DOT PRODUCT BETWEEN G AND a WHICH WILL GIVE A SCALAR TO ADD TO c**

****

**EVERY CONSTRAINT THEREFORE BASICALLY HAS ITS OWN NEURAL NET IT SEEMS**

I will say they do say this I think they more mean that they dont care about learning the transition function FOR STATE but they do care about learning how actions taken to modify the state (eg set value of a velocity) then affect the closeness to violation

We do not attempt to learn the full transition model, but solely the immediate-constraint functions ci(s, a).

BASICALLY THE SAFETY SIGNALS ARE THE VALUE OF SOMETHING LIKE TEMP OR LOCATION BUT WE DONT REALLY CARE ABOUT THAT WHAT WE CARE ABOUT IS THE TEMP AFTER AN ACTION THAT IS THE SAFETY SIGNAL ESTIMATORS g(s;w\_i) ^T SO BASICALLY In a way THE g IS LEARNING HOW THE DYNAMICS OF THE SYSTEM BRING A constraint closer or further from its boundary. NOT HOW THE OVER ALL STATE CHANGES UNLESS ONE OF THE STATE VARIABLES IS USED AS THE SAFETY SIGNAL(Position)

IMPORTANT

SO THE BEST WAY TO SEE THIS IS ITS NOT QUITE LEARNING THE DYNAMICS OF THE SYSTEM BUT HOW THE ACTIONS MAY BRING IT TO A STATE THAT IS NEAR VIOLATING A CONSTRAINT Because position IS THE VALUE USED FOR ENFORCING a constraint this is sometimes confusing

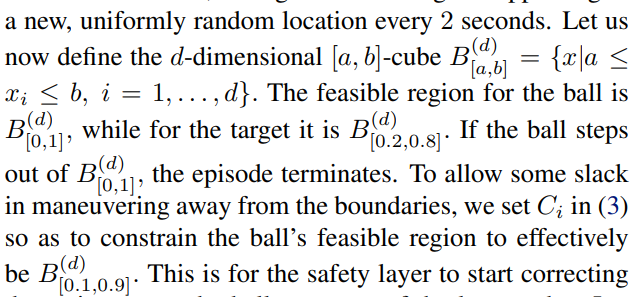
**Think about it like this position is the only relevant part of the state that actually affects the safety signal(how close are you to a boundary) but we control v however we have data that tells us a v of a certain value resulting in our position changing to be closer to a particular boundary. Thus the safety signal would grow closer to its limit.**

**When thinking about it in terms of a box where every corner is an obstacle. If you start in the center of the box and the action you set for a time step is to increase the velocity to the left (NOTE THE ACTION HAS TWO COMPONENTS x AND y IF WE ARE IN 2D)**

**Then you will grow closer to one boundary (left edge) but further from the right)**

where wi are weights of a NN, g(s; wi), that takes s as input and outputs a vector of the same dimension as a. This model is a first-order approximation to ci(s, a) with respect to a; i.e., an explicit representation of sensitivity of changes in the safety signal to the action using features of the state. See Fig. 1 for a visualization

Ok so f

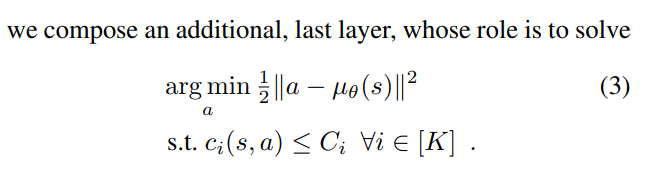


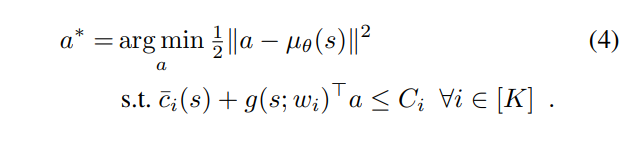
THEY DONT EXPLICITLY SAY IT BUT THESE ARE THE SAFETY SIGNALS

THIS IS THE DEF OF A SAFETY SIGNAL

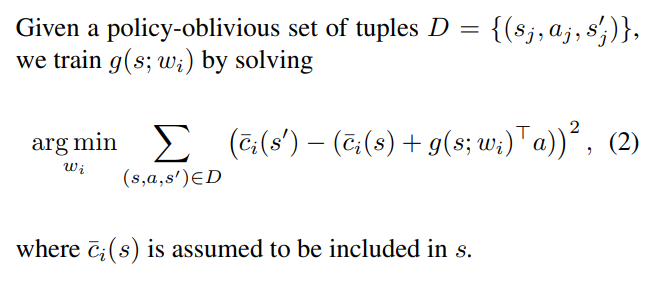
Based on that, we also define a set of safety signals C¯ = {c¯i : S → R | i ∈ [K]}. These are per-state observations of the immediate-constraint values, which we introduce for later ease of notation. To illustrate, if c1(s, a) is the temperature in a datacenter to be sensed after choosing a in s, c¯1(s’) is the same temperature sensed in s’ after transitioning to it.

Each safety signal c\_i(s, a) is approximated with a linear model with respect to a, whose coefficients are features of s, extracted with a NN.





IMPORTANT



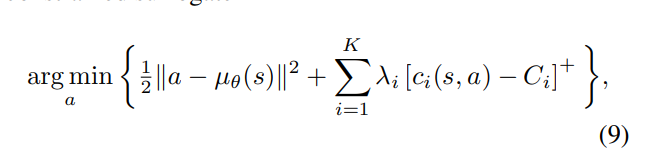
SAFETY SIGNAL FOR LEARNING IS INCLUDED IN D WHICH

Future work

Moreover, for other systems with multiple intersecting constraints, a joint model can be learned. For instance, instead of treating distances from two walls as two constraints, the minimum between them can be treated as a single constraint. In rudimentary experiments, this method produced similar results to not using it, when we grouped two constraints into one, in the first and simplest task in Section 7. However, jointly modeling the dynamics of more than a few safety signals with a single g(·; ·) network is a topic requiring careful attention, which we leave for future work

Training g(s; wi) on D is performed once per task as a pretraining phase that precedes the RL training. However, additional continual training of g(s; wi) during RL training is also optional. Since in our experiments continual training showed no benefit compared to solely pre-training, we only show results of the latter

Ok so this is the alternative formulation to to



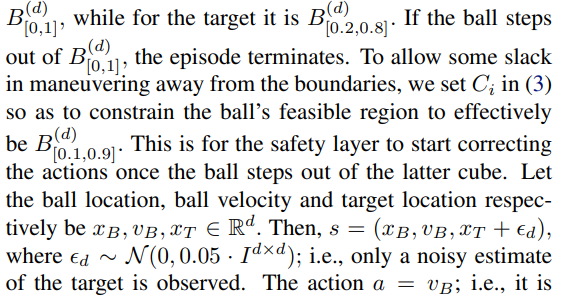
THIS IS NOT A SHIELDING PAPER ITS A CMDP lagrangian optimization paper but with a much more simple optimization

Safe exploration, as depicted above, traditionally requires access to data generated with some known behavior policy upon which gradual safe updates are performed; see (Thomas, 2015)

They compare multiple times but only at the end to reward shaping and apparently the WHOLE FUCKING POINT WAS IT AVOIDS TERMINATION DUE TO VIOLATION LIKE REWARD SHAPING

Apparently none of these papers give any fucking proofs of safety its just show that an experiment succeeded with zero collisions

GIVES STATE OF BALL



Complaints (points for future improvement from your point of view

Ngl this paper feels very stupid and contrived. Its very much a look we got zero constraint violations via experiments (no formal proof of safety tho because that's hard and we are going to criticize the papers who did have proofs of safety) ... Also that's only with a sufficient slack to boundaries on our constraint constants, ... and with deterministic transition function.... and with instant and complete control over the 1st derivative of the system we are controlling (and then saying that is a reasonable assumption for robots in the real world.....)

PAPER ITSELF

Do note throughout this they make a number of simplifying assumptions such as deterministic transitions (ddpg does not)

MAIN IDEA

So the idea is we have a pre-trained neural net that gives the coefficient of a linear model when given the state. We then use its output to correct the policy of the agent at each query the system makes. They allow for the use of pre recorded data that is off policy and single state transitions

They treat the threshold for constraints (temperature of a server or distance between robots obstacles above a margin) as what they call a safety signal (continuous scalar)

Thus, our safety layer is both differentiable and has a trivial three-line software implementation. Note that relating to our safety mechanism as a ‘safety layer’ is purely a semantical choice;

FIGURE 2

Our linearized safety-signal model allows a closed-form solution µ̃ θ (s) = arg min a f (s, a, µ(s)) that reduces to a trivial linear projection.

I think all they are saying by this is one of the benefits of a linear estimation of the phenomena is it allows us to do backprop through the safety layer

Note that relating to our safety mechanism as a

‘safety layer’ is purely a semantical choice; it merely is

a simple calculation that is not limited to the nowadays

popular deep policy networks and can be applied to any

continuous-control algorithm (not necessarily RL-based).

To summarize, this work is the first, to our knowledge, to

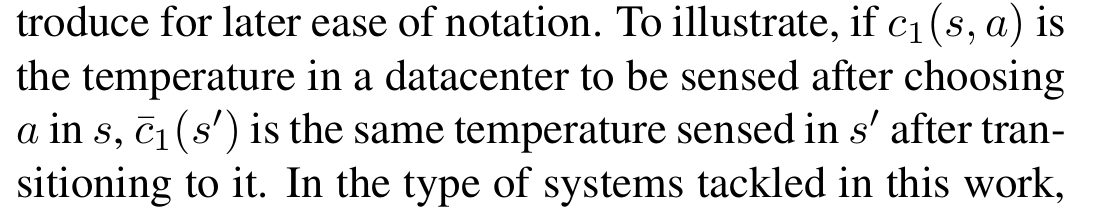
solve the problem of state-wise safety directly at the pol-

icy level, while also doing it in a data-driven fashion us-

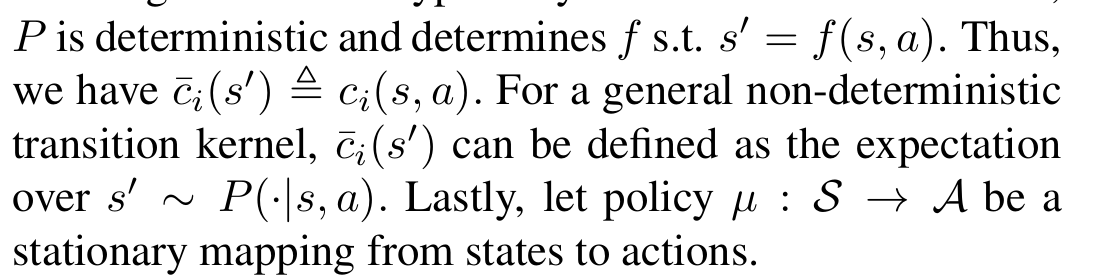
ing arbitrary data logs. Moreover, it can be applied to any

continuous-control algorithm; it is not restricted to a spe-

cific RL algorithm or any at all.



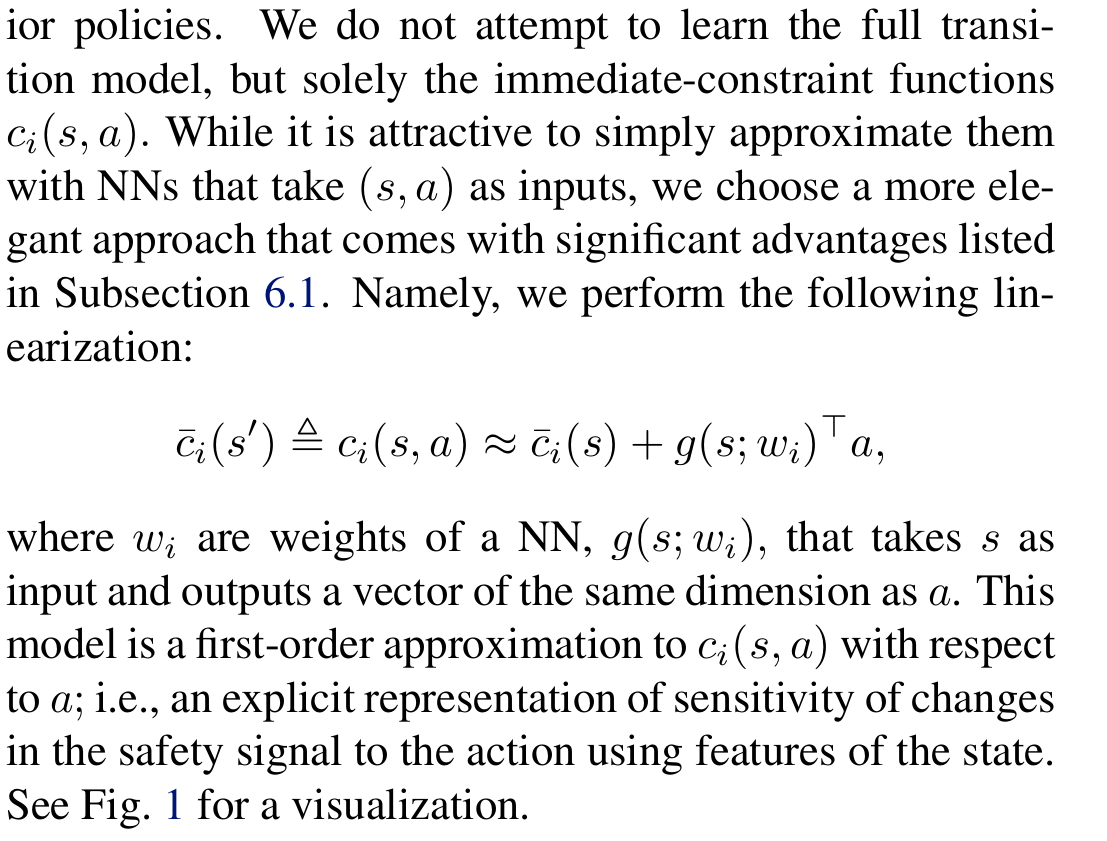
So c bar 1 is the safety signal and s is c1 is the immediate constraint function

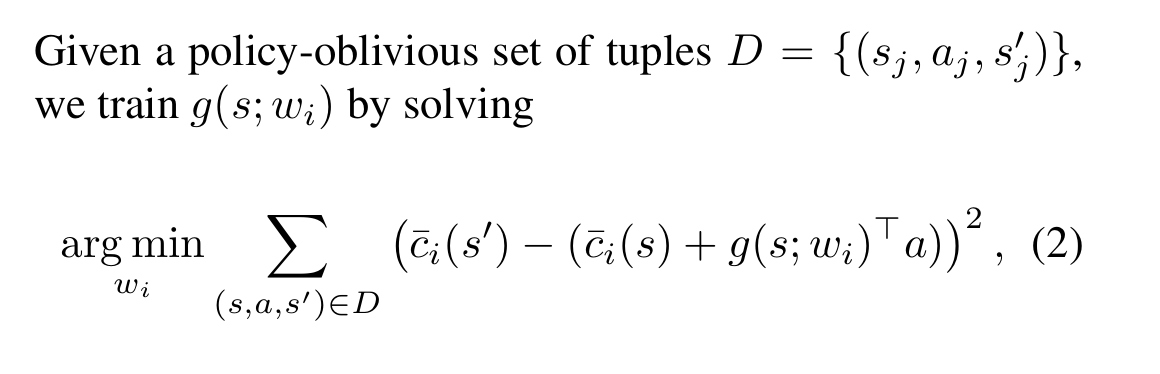


IMPORTANT

We stress that our goal is to ensure state-wise constraints not only for the solution of (1), but also for its optimization process. This goal might be intractable in general since for an arbitrary MDP some actions can have a long-term effect in terms of possible state paths. However, for the types of physical systems we consider it is indeed plausible that safety constraints can be ensured by adjusting the action in a single (or few) time step(s). In the context of our realworld use-cases, cooling system dynamics are governed by factors such as the first-order heat-transfer differential equation (Goodwine, 2010), and the second-order Newton differential equation that governs the water mass transfer. The latter also governs the movement of a robotic arm or a vehicle on which one applies forces. In these types of control problems, it is feasible to satisfy state-wise constraints even in the presence of inertia given reasonable slack in the choice of Ci . We expand on this further and provide evidence in Section 7.

We do not attempt to learn the full transition model, but solely the immediate-constraint functions c i (s, a).





Literally just we use the nn to estimate the coefficients of the linear approximation (change in value to each part of the state with respect to a for one timestep times the a. Basically the slope for one time unit) for our safety signals

In real-world applications such as the above, constraints are an integral part of the problem description, and never violating them is often a strict necessity. Therefore, in this work, we define our goal to be maintaining zero-constraint-violations throughout the whole learning process

However, jointly modeling the dynamics of more than a

few safety signals with a single g(·; ·) network is a topic re-

quiring careful attention, which we leave for future work.

Note that

accomplishing this goal for discrete action spaces is more

straightforward than for continuous ones. For instance, one

can pre-train constraint-violation classifiers on offline data

for pruning unsafe actions. However, in our context, this

goal becomes considerably more challenging due to the in-

finite number of candidate actions

Our approach relies on one-time initial pre-training of a

model that predicts the change in the safety signal over a

single time step. This model’s strength stems from its sim-

plicity: it is a first-order approximation with respect to the

action, where its coefficients are the outputs of a state-fed

neural network (NN). We then utilize this model in a safety

layer that is composed directly on top the agent’s policy to

correct the action if needed; i.e., after every policy query,

it solves an optimization problem for finding the minimal

change to the action such that the safety constraints are met.